

Physics-Guided Neural Networks for Distributed Sparse Gas Source Localization Using Poisson's Equation and Green's Function Method

Victor Scott Prieto Ruiz*, Dmitriy Shutin*, Thomas Wiedemann*[†] and Patrick Hinsin*

* German Aerospace Center (DLR), Muenchner Str. 20, 82234 Wessling, Germany

[†] Perception for Intelligent Systems, Munich Institute of Robotics and Machine Intelligence, TU Munich, Germany

Email: {victor.prietoruiz,dmitriy.shutin,thomas.wiedemann,patrick.hinsen}@dlr.de

Abstract—Finding sources or leaks of airborne material in Chemical, Biological, Radiological, or Nuclear (CBRN) accidents is crucial for effective disaster response. This paper makes use of sparse Bayesian learning (SBL) to cooperatively estimate source locations based on measurements by multiple robots or a sensor network. The SBL approach facilitates the identification of sparse source support, indirectly providing information about the number of sources and their locations. To achieve this, we introduce a novel method that includes a trained surrogated model for the gas dispersion process described by a Partial Differential Equation (PDE). Namely, a Physics-Guided Neural Network (PGNN) is employed to approximate a parameterized Green's function of the PDE. The obtained approximation is integrated into a gradient-based optimization process. The proposed method allows estimating super-resolution arbitrary source locations, eliminating constraints to a specific grid. Further, the newly proposed PGNN surrogate model comes with the advantage that the approach can be extended to cases where no analytic Green's function is available. Simulation results demonstrate the effectiveness of the proposed approach, showcasing its potential for enhanced airborne material detection in CBRN scenarios.

I. INTRODUCTION

Accurate modeling and estimation of airborne released material are essential for response to chemical, biological and radiological accidents. Autonomous mobile robotic platforms, equipped with appropriate sensors, are ideal tools to explore and gather necessary information in such environments, "learning" their environment as needed to derive optimal operation/control strategies.

In general, the physics governing the spatio-temporal evolution of dispersed materials is effectively captured by PDEs [1]. For example, the non-homogeneous scalar transport equation can simulate the propagation of material from various emitting sources into the air. However, for exploring robots to have prior domain knowledge, they should determine unknown model parameters, such as source locations and magnitudes. These can then be estimated from sensor data.

In this paper we tackle the Gas Source Localization (GSL) problem [2], the goal being to estimate the number and locations of multiple gas sources. We employ a Poisson equation as a model for steady-state gas dispersion, offering a simplified illustration of the proposed methodology that can be extended to more realistic models.

Traditionally, robot movements in GSL strategies relied on techniques such as *chemotaxis* and *anemotaxis* [2]. However, these approaches often assume a known number of sources and smooth concentration gradients. Model-based methods can have more nuance and build upon the mathematical structure of the process model, treating sources as unknown parameters. This allows for modeling uncertainties, using different models [3] and handling unknown environments [4].

Infotaxis-based methods further enhance model-based GSL by autonomously guiding robots to sources using information-theoretic criteria (see e.g., [5]). Despite these advancements, most GSL research assumes a fixed and known number of sources due to the computational challenges of integer optimization.

However, recent works introduced a combination of SBL [6] and a PDE-based dispersion modeling, relaxing the fixed-source-count assumption [7]. These approaches assume a source distribution to be sparse, thus indirectly estimating both the number and locations of the sources in simulations [7] and in real experiments [8].

One limitation of this approach is the discretization of the PDE, which restricts the source locations to mesh elements and increases the number of unknown parameters to estimate. To address this deficiency, in [9] we developed an algorithm relying on the Green's function method; in [10] the method was extended to a decentralized setting for solving the corresponding GSL problem over a network of agents.

In this research, we expand upon our previous work and involve the approximation of Green's function through the use of a neural network surrogate. Specifically, the network is trained under constraints in the form of a PDE, characterizing a type of network commonly referred to as a PGNN.

This approach is helpful because it can be applied to various types of PDEs with different boundary conditions and the generation of smooth approximations. In addition, the computationally expensive task of training the Green's function surrogate can be done offline – especially useful when no analytical formulations of Green's function exist – leaving only the task of gradient computation with respect to the network output. Notably, it accommodates Green's functions with discontinuities – a crucial feature for algorithms reliant on derivatives of these functions.

II. SIGNAL MODEL

Consider GSL over some d -dimensional exploration area Ω . We assume that the spatial gas concentration at equilibrium over Ω can be described by a time-invariant linear diffusion PDE (also known as Poisson's equation) in the following form:

$$-\kappa \Delta f(\mathbf{x}) = \sum_{l=1}^L w_l \delta_{\theta_l}(\Omega), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d \quad (1)$$

$$\text{s.t. } f(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega, \quad (2)$$

where κ is a diffusion coefficient, $f(\mathbf{x})$ is a spatial gas concentration intensity, and Δ is a Laplace operator. The right-hand side (RHS) of (1) is a superposition of L gas sources – Dirac measures over Ω with a support $\theta_l \in \Omega$, and having release rates w_l , $l \in \mathcal{L} \triangleq \{1, \dots, L\}$. Equation (1) is augmented with a Dirichlet boundary condition (2) at the boundary $\partial\Omega$.

We assume only a few gas sources are present, such that w_l , $l \in \mathcal{L}$ are sparse; i.e., not all possible gas sources are active, and most $w_l = 0$. Furthermore, we will select L to exceed the true number of sources, which is sometimes referred to as *max-search* approach [11]. The model (1) and (2) thus represents a spatial distribution of the gas released from several (constant) sources in the absence of wind.

Assume for a moment that both w_l and θ_l , $l \in \mathcal{L}$ are known. $f(\mathbf{x})$ can then be obtained by solving (1) subject to (2). We use here a method of Green's functions [1], which determines $f(\mathbf{x})$ as

$$f(\mathbf{x}) = \int_{\Omega} G(\mathbf{x}, \boldsymbol{\theta}) \sum_{l=1}^L w_l \delta_{\theta_l}(\Omega) d\boldsymbol{\theta} = \sum_{l=1}^L w_l G(\mathbf{x}, \boldsymbol{\theta}_l), \quad (3)$$

where $G(\mathbf{x}, \boldsymbol{\theta})$ is a Green's function defined as a solution to

$$-\kappa \Delta G(\mathbf{x}, \boldsymbol{\theta}) = \delta_{\boldsymbol{\theta}}(\Omega), \quad \mathbf{x}, \boldsymbol{\theta} \in \Omega, \quad (4)$$

subject to the constraint $G(\mathbf{x}, \boldsymbol{\theta}) = 0$, $\mathbf{x} \in \partial\Omega$; i.e., $G(\mathbf{x}, \boldsymbol{\theta})$ is a 'response' of the PDE to a single source at $\boldsymbol{\theta}$.

Now, consider a discretization of (1) where Ω is partitioned into N not necessarily regular grid cells. For each cell with center coordinates \mathbf{x}_i , $i \in \mathcal{N} \triangleq \{1, \dots, N\}$, we then assume a constant concentration value $f(\mathbf{x}_i) = \text{const}$. The corresponding concentrations are then aggregated into a vector \mathbf{f} with i th element given by $[\mathbf{f}]_i = f(\mathbf{x}_i)$. Similarly, we define an $N \times L$ matrix $\mathbf{G}(\boldsymbol{\Theta})$ with i, j element $[\mathbf{G}(\boldsymbol{\Theta})]_{i,j} = G(\mathbf{x}_i, \boldsymbol{\theta}_j)$, $i \in \mathcal{N}$, $j \in \mathcal{L}$, where $\boldsymbol{\Theta} \triangleq [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_L]$. This allows us to rewrite (3) in a matrix-vector form as

$$\mathbf{f} = \mathbf{G}(\boldsymbol{\Theta})\mathbf{w} \quad (5)$$

with $\mathbf{w} \triangleq [w_1, \dots, w_L]^T$. Let us point out that while we discretized Ω , the location of the sources in the model (5) are not restricted to this grid and can take arbitrary values.

Consider now a network of K robotic agents. We will model the network with a strongly connected, weighted graph. Assume now that each agent collects M_k , $k \in \mathcal{K} \triangleq \{1, \dots, K\}$, noisy samples of the concentration $f(\mathbf{x})$ at locations $\mathbf{x}_{m,k}$, $m = 1, \dots, M_k$. Furthermore, w.l.o.g. we assume that $N \gg$

M_k and that $\forall m, k$, $\mathbf{x}_{m,k}$, are a subset of discretization cells¹. The measurements of the agent k are collected in a vector $\mathbf{z}_k \in \mathbb{R}^{M_k}$ such that

$$\mathbf{z}_k = \Phi_k \mathbf{f} + \boldsymbol{\xi}_k, \quad k \in \mathcal{K}, \quad (6)$$

where $\Phi_k \in \mathbb{R}^{M_k \times N}$ is a 0-1 selection matrix that indicates measured elements of \mathbf{f} . The perturbation $\boldsymbol{\xi}_k$ is assumed to be homoscedastic zero-mean Gaussian with precision matrix $\lambda_{\xi} \mathbf{I}$ for $\lambda_{\xi} > 0$.

Our goal now is to use $\mathbf{z} \triangleq [\mathbf{z}_1^T, \dots, \mathbf{z}_K^T]^T$ to cooperatively estimate a sparse vector \mathbf{w} , locations $\boldsymbol{\Theta}$, and recover \mathbf{f} from (5). To this end, we pursue a Bayesian approach for parameter estimation.

Specifically, we consider the following posterior probability density function (pdf) of the variables of interest:

$$p(\mathbf{f}, \mathbf{w}, \boldsymbol{\Theta} | \mathbf{z}) \propto p(\mathbf{z} | \mathbf{f}) p(\mathbf{f} | \mathbf{w}, \boldsymbol{\Theta}) p(\mathbf{w}) p(\boldsymbol{\Theta}), \quad (7)$$

where source locations $\boldsymbol{\theta}$ and their rates \mathbf{w} are assumed to be independent. Based on (6) we see that $p(\mathbf{z} | \mathbf{f}) = \prod_{k \in \mathcal{K}} \mathcal{N}(\mathbf{z}_k | \Phi_k \mathbf{f}, \lambda_{\xi}^{-1} \mathbf{I})$ is a Gaussian likelihood function. The pdf $p(\mathbf{f} | \mathbf{w}, \boldsymbol{\Theta})$ encodes the deterministic relationship between sources and concentration \mathbf{f} following (5), and is modeled with a Dirac distribution $p(\mathbf{f} | \mathbf{w}, \boldsymbol{\Theta}) = \delta_{\mathbf{G}(\boldsymbol{\Theta})\mathbf{w}}(\mathbb{R}^N)$ over N -dimensional space \mathbb{R}^N .

Concerning the prior $p(\boldsymbol{\Theta})$ we will assume it to be uniform over Ω , i.e., $p(\boldsymbol{\Theta}) \propto \text{const}$. In case of $p(\mathbf{w})$ we instead employ a modeling approach used in SBL [6].

Formally this implies a hierarchical factorable prior $p(\mathbf{w} | \boldsymbol{\gamma}) p(\boldsymbol{\gamma}) = \prod_{l=1}^L p(w_l | \gamma_l) p(\gamma_l)$, where $p(w_l | \gamma_l) = \mathcal{N}(w_l | 0, \gamma_l)$, $l \in \mathcal{L}$ [6]. For each $l \in \mathcal{L}$ the hyperparameter $\gamma_l \geq 0$, also called sparsity parameter, regulates the "width" of $p(w_l | \gamma_l)$. When $\gamma_l \rightarrow 0$, the prior $p(w_l | \gamma_l)$ collapses to a Dirac measure centered at the origin, driving the posterior estimate of w_l towards 0.

The hyperpriors $p(\gamma_l)$, $l \in \mathcal{L}$, are selected as uniform, i.e. $p(\gamma) \propto 1$ [12], which leads to so-called evidence maximization procedures for estimation of hyperparameters [6]. Now, the resulting posterior pdf becomes

$$p(\mathbf{f}, \mathbf{w}, \boldsymbol{\gamma}, \boldsymbol{\Theta} | \mathbf{z}) \propto p(\mathbf{z} | \mathbf{f}) p(\mathbf{f} | \mathbf{w}, \boldsymbol{\Theta}) p(\mathbf{w}, \boldsymbol{\gamma}), \quad (8)$$

recalling that $p(\boldsymbol{\Theta})$ was uniform as well. In what follows we propose a distributed optimization algorithm to iteratively maximize (8).

III. DISTRIBUTED SUPERRESOLUTION GSL WITH NEURAL NETWORK SURROGATE

The the following we briefly summarize the distributed superresolution GSL algorithms, referring the reader to [10] for more details. We then focus on the neural network surrogate learning to be used in the proposed algorithm.

¹As discretization of Ω is arbitrary, we can always select it to include measurement locations

A. Distributed Superresolution GSL

We begin with the estimation of γ by treating Θ fixed at $\hat{\Theta}$ and marginalize (8) over \mathbf{f} , which leads to

$$p(\mathbf{w}, \gamma, \hat{\Theta} | \mathbf{z}) = \int p(\mathbf{f}, \mathbf{w}, \gamma, \hat{\Theta} | \mathbf{z}) d\mathbf{f} \propto p(\mathbf{z} | \mathbf{w}, \hat{\Theta}) p(\mathbf{w}, \gamma)$$

By keeping Θ constant, we linearize the problem, aligning the resulting posterior with the one used in a conventional SBL framework [6]. This enables us to utilize established algorithms for maximizing this posterior. Specifically, we'll use a distributed implementation of the Fast Marginal Likelihood Maximization (FMLM) algorithm from [13] for the maximization process. Our specialized modifications to the algorithm are outlined below (see also [10] for more details). We define

$$\bar{\Phi} \triangleq \lambda_{\xi} \sum_{k \in K} \Phi_k^T \Phi_k, \quad \bar{\mathbf{z}} \triangleq \lambda_{\xi} \sum_{k \in K} \Phi_k^T \mathbf{z}_k \quad (9)$$

$$\mathbf{D} \triangleq \mathbf{G}(\hat{\Theta})^T \bar{\Phi} \mathbf{G}(\hat{\Theta}), \quad \text{and} \quad \mathbf{d} \triangleq \mathbf{G}(\hat{\Theta})^T \bar{\mathbf{z}}. \quad (10)$$

where $\bar{\Phi}$ and $\bar{\mathbf{z}}$ are computed distributively using an averaged consensus algorithm [14]. Using the latter, and given an estimate of $\hat{\gamma}$, the weight posterior $p(\mathbf{w} | \hat{\gamma}, \hat{\Theta}, \mathbf{z}) \propto p(\mathbf{z} | \mathbf{w}, \hat{\Theta}) p(\mathbf{w} | \hat{\gamma})$ can be shown to be Gaussian with the covariance matrix and the mean given by

$$\hat{\Sigma}_{\mathbf{w}} = (\mathbf{D} + \hat{\Gamma}^{-1})^{-1} \quad \text{and} \quad \hat{\mathbf{w}} = \hat{\Sigma}_{\mathbf{w}} \mathbf{d} \quad (11)$$

respectively, where $\hat{\Gamma} \triangleq \text{diag}(\hat{\gamma})$. The estimate of $\hat{\gamma}$ is then found as a minimizer of

$$\begin{aligned} C(\gamma) &\triangleq -\log p(\mathbf{z} | \gamma, \hat{\Theta}) = -\log \int p(\mathbf{z} | \mathbf{w}, \hat{\Theta}) p(\mathbf{w} | \gamma) d\mathbf{w} \\ &= \frac{1}{2} \log |\Sigma_{\gamma}(\hat{\Theta})| + \frac{1}{2} \mathbf{z}^T \Sigma_{\gamma}(\hat{\Theta})^{-1} \mathbf{z} \end{aligned} \quad (12)$$

where $\Sigma_{\gamma}(\hat{\Theta}) \triangleq \lambda_{\xi}^{-1} \mathbf{I} + \bar{\Phi} \mathbf{G}(\hat{\Theta}) \Gamma \mathbf{G}(\hat{\Theta})^T \bar{\Phi}^T$, $\bar{\Phi} \triangleq [\Phi_1^T, \dots, \Phi_K^T]^T$, and $\Gamma \triangleq \text{diag}(\gamma)$.

The estimation of location parameters Θ with fixed support estimate $\hat{\gamma}$ is achieved by maximizing the posterior $p(\mathbf{f}, \hat{\mathbf{w}}, \hat{\gamma}, \Theta | \mathbf{z})$ with respect to Θ . This is equivalent to the following optimization:

$$\begin{aligned} \min_{\Theta} \quad & \left\{ J(\Theta) \triangleq -\log p(\mathbf{z} | \mathbf{f}) = \frac{\lambda_{\xi}}{2} \|\mathbf{z} - \bar{\Phi} \mathbf{f}\|^2 \right\} \quad (13) \\ \text{s.t.} \quad & g(\mathbf{f}, \Theta) \triangleq \mathbf{f} - \mathbf{G}(\Theta) \mathbf{w} = \mathbf{0} \end{aligned}$$

where both \mathbf{f} and \mathbf{w} are implicit functions of Θ . In [10] we showed that this nonlinear optimization can be solved with a combine-then-adapt (CTA) diffusion strategy [15] with the following update iterations for each agent k in the network:

$$\Psi_k = \sum_{l \in \mathcal{N}(k)} \alpha_{lk} \hat{\Theta}_l^{[j]}, \quad (\text{combination}) \quad (14)$$

$$\hat{\Theta}_k^{[j+1]} = \Psi_k - \mu \nabla_{\Theta} J(\Theta) |_{\Theta=\Psi_k} \quad (\text{adaptation}). \quad (15)$$

where α_{lk} , $k, l \in \mathcal{K}$, are graph edge weights, $\nabla_{\Theta} J(\Theta) = (\bar{\mathbf{z}} - \bar{\Phi} \mathbf{G}(\Theta) \mathbf{w})^T \frac{\partial g(\mathbf{f}, \Theta)}{\partial \Theta}$ is a gradient of (13) evaluated at Ψ_k and corresponding \mathbf{w} given by (11),

$$\frac{\partial g(\mathbf{f}, \Theta)}{\partial \theta_l} = - \left[w_l \frac{\partial G(\mathbf{x}_0, \theta_l)}{\partial \theta_l}, \dots, w_l \frac{\partial G(\mathbf{x}_{N-1}, \theta_l)}{\partial \theta_l} \right]^T,$$

and μ is an appropriately chosen step size [9], [10].

Evaluating the gradient necessitates a derivative of the Green's function $G(\mathbf{x}, \theta)$ with respect to θ . While the Green's function can sometimes be analytically computed, as in the case of the Poisson equation discussed here, this is not always feasible. Additionally, Green's functions can be non-smooth and/or discontinuous, potentially causing the second derivative of (13) to be unbounded in certain parameter space locations, potentially leading to divergence of gradient updates (15).

To address this, we approximate the Green's function of the Poisson PDE with a PGNN, ensuring a smooth representation across the parameter domain. The specifics of this approach are outlined below.

B. Neural Network Surrogate

Because the model surrogate approximates Green's function $G(\mathbf{x}, \theta)$, we will denote our approximation by $\tilde{G}(\mathbf{x}, \theta)$ throughout this section.

1) *Network Architecture*: The surrogate model we chose was a fully-connected Multilayer Perceptron (MLP). A popular configuration for a D -layer MLP $\mathcal{P}(\mathbf{y}) : n_{\text{in}} \rightarrow n_{\text{out}}$ is defined for an input $\mathbf{y} \in \mathbb{R}^{n_{\text{in}}}$ by:

$$\begin{aligned} \mathbf{y}_0 &\triangleq \mathbf{y}, \mathbf{y}_k \triangleq \rho(\mathbf{W}_k \mathbf{y}_{k-1} + \mathbf{b}_k), k \in [1, \dots, M-1] \\ \mathcal{P}(\mathbf{y}) &\triangleq \mathbf{W}_D \mathbf{y}_{D-1} + \mathbf{b}_D, \end{aligned} \quad (16)$$

where $\mathbf{W}_k \in \mathbb{R}^{n_k \times n_{k-1}}$, $\mathbf{b}_k \in \mathbb{R}^{n_k}$ are the k -th layer's weight matrix and bias vector, respectively, with $n_0, \dots, n_D \in \mathbb{Z}_+$ the number of neurons at each layer ($n_0 = n_{\text{in}}, n_D = n_{\text{out}}$); and $\rho : \mathbb{R} \rightarrow \mathbb{R}$ is the nonlinear activation function, applied componentwise to each vector element. We note that the last network layer is a linear scaling layer without an activation function, as it is common.

Our goal is to approximate Green's function $G(\mathbf{x}, \theta)$ parameterized by θ . The input of the network is therefore the concatenation of $[\mathbf{x}, \theta] \in \Omega \times \Omega$, and its output is a real number. This sets $n_{\text{in}} = n_0 = 2d, n_{\text{out}} = n_D = 1$.

2) *Enforcing Boundary Conditions*: To ensure the NN satisfies the Boundary Conditions (BCs), we use a hard-enforcement approach, suggested in [16, Sec. 5.1.1.]. This enforces homogeneous Dirichlet BCs exactly, by multiplying the output of a MLP by a smooth cutoff function $\psi : \Omega \rightarrow \mathbb{R}$, which is zero on the boundary $\delta\Omega$:

$$\tilde{G}(\mathbf{x}, \theta) \triangleq \mathcal{P}([\mathbf{x}, \theta]) \psi(\mathbf{x}) \quad (17)$$

with square brackets $[\cdot, \cdot]$ denoting vector concatenation. We used the R - m -based approximate distance function to the two endpoints in [16] as ψ .

3) *Network Training*: To train the network and determine its weights, we used a physics-guided approach, following the nomenclature in [17]. This supervised learning method amounts to sampling the network output at random sampling points in our problem domain and comparing it to a reference value. We define the *Physics-Guided H1 Loss* as follows:

$$L_{\text{PG}} \triangleq \sum_{i=1}^{N_t} \left[(\tilde{G}(\mathbf{x}_i, \boldsymbol{\theta}_i) - G(\mathbf{x}_i, \boldsymbol{\theta}_i))^2 + \|\nabla_x \tilde{G}(\mathbf{x}_i, \boldsymbol{\theta}_i) - \nabla_x G(\mathbf{x}_i, \boldsymbol{\theta}_i)\|_2^2 \right], \quad (18)$$

where \tilde{G} is our surrogate model, G is the (analytical) reference Green's function, ∇_x is the gradient of the Green's function with respect to its first argument, and $\mathbf{x}_i, \boldsymbol{\theta}_i \in \Omega$ represent N_t uniform-randomly sampled points of evaluation for the Green's function model.

The H1-loss differs from other physics-guided approaches in that it additionally gives the network information about the gradient of the reference solution. This is done with the intention of making the surrogate model's gradients more useful for SBL.

IV. SIMULATION RESULTS

We now provide some simulation results to demonstrate the performance of the method. For simplicity, we will consider a Poisson equation in 1D, setting $\Omega = [0, 1]$ and $\kappa = 1$. In this case the Green's function has analytical solution $G(x, \theta) = x(1 - \theta)$ for $0 \leq x \leq \theta$ and $G(x, \theta) = -\theta(x - 1)$ for $\theta < x \leq 1$.

We generate 3 sources with rates set to 1 and locations selected as $\theta_1 = 0.2 + 0.05\epsilon_1$, and $\theta_i = \theta_{i-1} + 0.1 + 0.2\epsilon_i$, $i = 2, 3$, where ϵ_i , $i = 1, \dots, 3$ are uniformly and independently drawn from a unit interval. Such source generation ensures their separability.

For the distributed GSL algorithm we varied M , the number of measurements, and equally split them between $K = 6$ agents. The used connectivity graph is a geometric graph with 50% connectivity; we select the weights α_{kl} , $k, l \in \mathcal{K}$, according to the Laplacian rule [15].

The neural network Green's function surrogate used 5 layers, with 30 neurons in each hidden layer ($n_1, \dots, n_4 = 30$). The network used the softplus activation function. We trained and implemented the network in Python using the Pytorch², Numpy³ and Scipy⁴ libraries. We used the ADAM Optimizer [18] with a standard learning rate $l = 10^{-3}$. To train the network, we randomly resampled the $N_t = 15000$ sampling points for $(x, \theta) \in [0, 1] \times [0.1, 0.9]$ at each iteration, making the loss function stochastic. We trained the network for 15000 iterations.

²A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* 32, Curran Associates, Inc., 2019, pp. 8024–8035.

³C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020.

⁴P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

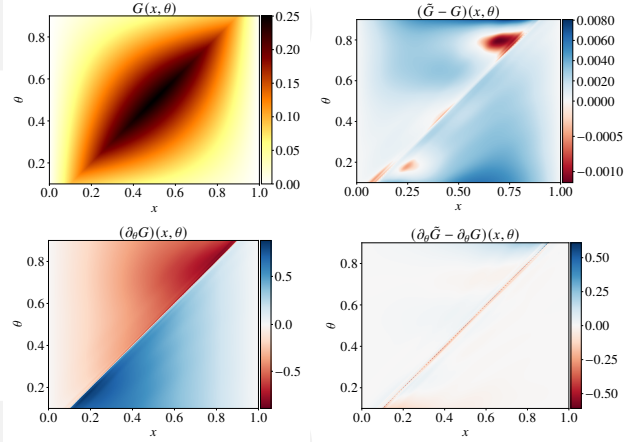


Fig. 1: Surrogate Model Performance Representation.

The top and bottom row show the model's performance of approximating function values, and derivative with respect to θ , respectively. For each row, the left plot represents the network values and the right plot the difference between the network and reference values. The color plots have different scales for positive and negative values.

We briefly compare the trained neural network output with the reference solution. In Fig. 1 we observe that the network closely follows the reference over the whole domain. Values deviate the most when θ is close to 0.1 or 0.9. We see that when looking at the derivative of the network with respect to the source location, we observe oscillations, particularly in areas close to the discontinuity present at the source location. These oscillations result from the continuous network approximating the discontinuity in the derivative and could impact the performance of the SBL approach.

We now compare the proposed distributed GSL (DSR-GSL) and its version using an NN surrogate (DSR-NN-LL) with two benchmark strategies, each using the exact derivative of the Green's Function: its centralized version (C-GSL), and a centralized re-weighted LASSO (rLASSO) [12], [19]. For rLASSO we linearize the model by discretizing Ω into $5N$ points; as estimated sources, we keep only the estimated weights with magnitudes $> 10^{-5}$, in order to compensate for the numerics of the solver.

As metrics, we use the estimated number of components \hat{L} , the MSE between the estimated and true concentration \mathbf{f} , and true and false positive rates of source detections. Detections were discretized cell-wise, that is, a source was considered to be detected if the localization algorithm placed a source in the same cell as the ground truth. The cells were chosen to be even-width of dimension $\frac{1}{N}$. For example, in the provided sample run (cf. Fig. 2), since $N = 100$, the middle source was not detected and there is a false positive adjacent to it.

The results, averaged over 400 random runs, are shown in Fig. 3 for SNR=5dB (low) and SNR=30dB (high) regimes as functions of M . As we see in Fig. 3, the centralized algorithms seem to introduce "artifacts" – small in amplitude (as seen from MSE plots), yet detected as false sources (cf.

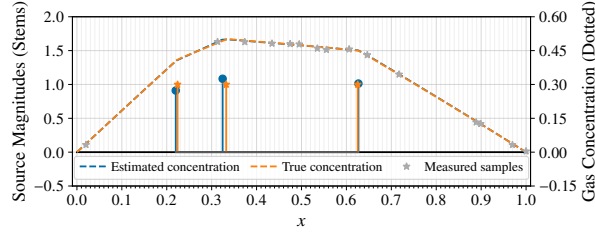


Fig. 2: Sample GSL run of an agent in DSR-NN., $M = 90$. Stems: Source Location and Magnitudes; Lines: Concentration Values; Points: Measured Samples from this agent.

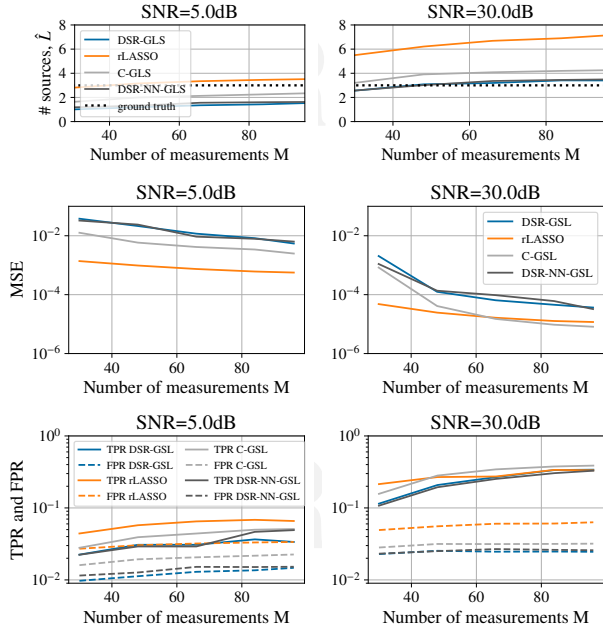


Fig. 3: GSL Performance Metrics. Top: Estimated \hat{L} ; Middle: Concentration MSE; Bottom: true positive rate (TPR) and false positive rate (FPR).

number of sources plot).

This is more pronounced for rLASSO, which seems to overfit the data as confirmed by the MSE and higher FPR. DSR-GSL and C-GSL remain more conservative in this respect, with DSR-GSL producing fewer artifacts, losing a bit in MSE, but achieving similar TPR to the other methods, especially in high SNR regime. The discrepancy in MSE between DSR-GSL and C-GSL in high SNR regime is likely attributed to slight variations in location estimates between agents due to the CTA rule.

Overall we can see that the NN-based distributed GSL method performs very comparably to the DSR-GSL, despite not having a true representation of Green's function. This gives an indication that the gradient of the PGNN approximation is suitable for use in optimization.

V. CONCLUSION

The results have shown that it is possible to use SBL to solve the GSL problem without requiring an analytical Green's

function. In our case, the network surrogate did not seem to negatively impact the results. Because the model still performs well with a network surrogate, we have shown that it is possible to implement this algorithm without computationally-expensive PDE solvers on the robots. Future work could include an extension to more complicated distributed implementations in two or three dimensions.

REFERENCES

- [1] P. Kythe, *Green's Functions and Linear Differential Equations: Theory, Applications, and Computation* (Chapman & Hall/CRC Applied Mathematics & Nonlinear Science). CRC Press, 2011.
- [2] A. Francis, S. Li, C. Griffiths, and J. Siem, "Gas source localization and mapping with mobile robots: A review," *Journal of Field Robotics*, pp. 1341–1373, May 2022.
- [3] J. R. Bourne, E. R. Pardyjak, and K. K. Leang, "Coordinated bayesian-based bioinspired plume source term estimation and source seeking for mobile robots," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 967–986, 2019.
- [4] C. Rhodes, C. Liu, P. Westoby, and W.-H. Chen, "Autonomous search of an airborne release in urban environments using informed tree planning," *Autonomous Robots*, vol. 47, no. 1, pp. 1–18, Jan. 2023.
- [5] M. Vergassola, E. Villermaux, and B. I. Shraiman, "infotaxis" as a strategy for searching without gradients," *Nature*, vol. 445, pp. 406–409, 7126 2007.
- [6] M. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Machine Learning Res.*, vol. 1, pp. 211–244, Jun. 2001.
- [7] T. Wiedemann, C. Manss, and D. Shutin, "Multi-agent exploration of spatial dynamical processes under sparsity constraints," *Autonomous Agents and Multi-Agent Systems*, Jul. 2017.
- [8] T. Wiedemann, D. Shutin, and A. Lilienthal, "Experimental validation of domain knowledge assisted robotic exploration and source localization," in *IEEE International Conference on Autonomous Systems (ICAS)*, Oct. 2021.
- [9] D. Shutin, T. Wiedemann, and P. Hinsin, "Detection and estimation of gas sources with arbitrary locations based on poisson's equation," *IEEE Open Journal of Signal Processing*, vol. 5, pp. 359–373, 2024.
- [10] D. Shutin, T. Wiedemann, and P. Hinsin, "Distributed super-resolution gas source localization based on poisson equation," in *9th IEEE Conf. Computational Advances in Multisensor Adaptive Processing*, to appear, 2003.
- [11] P.-J. Chung, "A max-search approach for DOA estimation with unknown number of signals," *IEEE J. Sel. Topics in Signal Process.*, vol. 4, no. 3, Jun. 2010.
- [12] D. Wipf and S. Nagarajan, "A new view of automatic relevance determination," in *Proc. 21 Annual Conf. Neural Information Processing Systems*, Vancouver, British Columbia, Canada: MIT Press, Dec. 2007.
- [13] C. Manss, D. Shutin, and G. Leus, "Consensus based distributed sparse bayesian learning by fast marginal likelihood maximization," *IEEE Signal Processing Letters*, vol. 27, pp. 2119–2123, 2020.
- [14] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, IEEE, 2005, pp. 63–70.
- [15] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.
- [16] N. Sukumar and A. Srivastava, "Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 389, p. 114 333, 2022.
- [17] S. A. Faroughi, N. Pawar, C. Fernandes, S. Das, N. K. Kalantari, and S. K. Mahjour, "Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing," *arXiv preprint arXiv:2211.07377*, 2022.
- [18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.
- [19] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *J. Roy. Statist. Soc.*, vol. 58, pp. 267–288, 1996.